Periodic Table of Agile Principles and Practices

by Jerome Kehrli

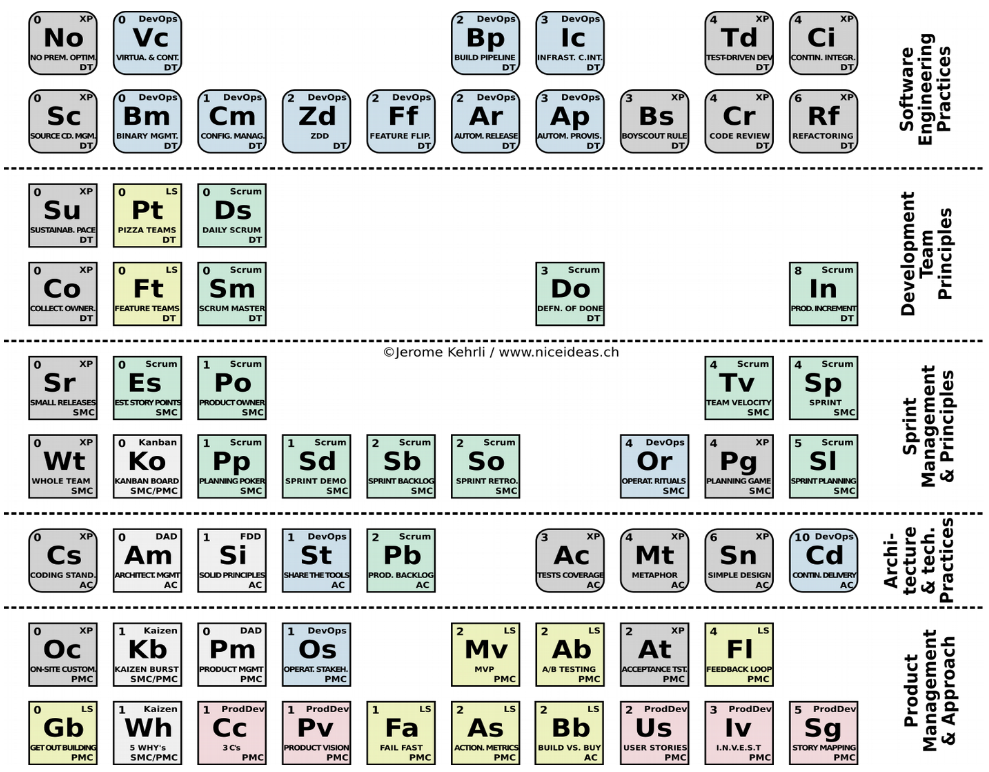
Written in June, 2017

After writing my previous article, I wondered how I could represent on a single schematic all the *Agile Principles and Practices* from the methods I am following, XP, Scrum, Lean Startup, DevOps and others.
I found that the approach I used in in a former schematic - a graph of relationship between practices - is not optimal. It already looks ugly with only a few practices and using the same approach for the whole set of them would make it nothing but a mess.

So I had to come up with something else, something better.
Recently I fell by chance on the Periodic Table of the Elements... Long time no see...
Remembering my physics lessons in University, I always loved that table. I remembered spending hours understanding the layout and admiring the beauty of its natural simplicity.
So I had the idea of trying the same layout, not the same approach since both are not comparable, really only the same layout for *Agile Principles and Practices*.
The result is hereunder.

Table of Contents

1. The Periodic Table of Agile Principles and Practices



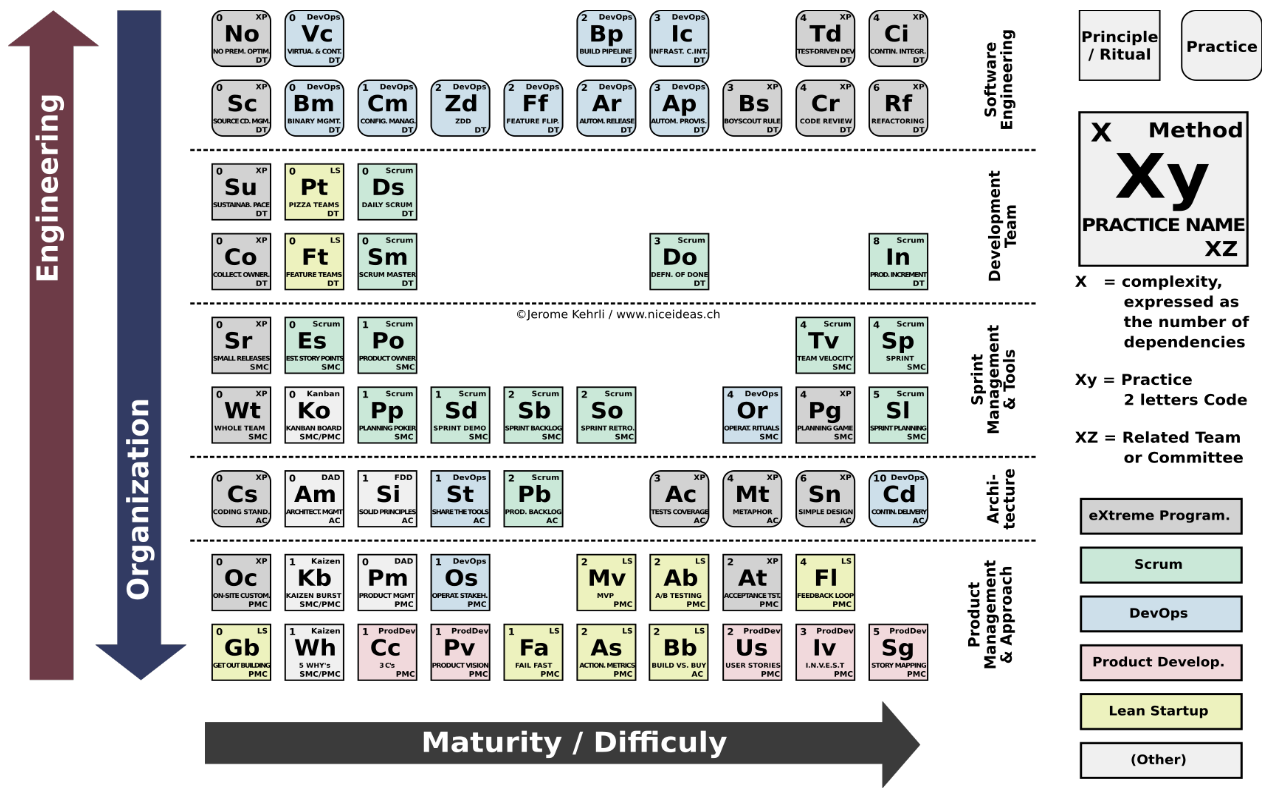
The layout principle is and the description of the principles and practices is explained hereafter.

2. Layout Principle

- The **origin Method** such as XP, Scrum, DevOps, etc is indicated by the color as well as the name of the method on the top-right corner.
- The **category**, *Principle* or *Practice* is indicated by the shape: rectangle or round corners.
- The number represents the **complexity** expressed as the number of dependencies.
- The **team or committee** concerned with the principle or practice is indicated as note on the bottom-right corner.
- The **horizontal dimension** is related to the complexity. The more on the right is an element, the higher its complexity.

- The **vertical dimension** is related to classifying principles and practices more organization or more related to engineering, in specific layers related to the category or team they apply to.

This is best presented as follows:



Engineering
Organization
0 XP No NO PREM. OPTIM. DT
0 DevOps Vc VIRTUA. & CONT. DT
2 DevOps Bp BUILD PIPELINE DT
3 DevOps Ic INFRAST. C.INT. DT
4 XP Td TEST-DRIVEN DEV DT
4 XP Ci CONTIN. INTEGR. DT
0 XP Sc SOURCE CD. MGM. DT
0 DevOps Bm BINARY MGMT. DT
1 DevOps Cm CONFIG. MANAG. DT
2 DevOps Zd ZDD DT
2 DevOps Ff FEATURE FLIP. DT
2 DevOps Ar AUTOM. RELEASE DT
3 DevOps Ap AUTOM. PROVIS. DT
3 XP Bs BOYSCOUT RULE DT
4 XP Cr CODE REVIEW DT
6 XP Rf REFACTORING DT
Software Engineering
0 XP Su SUSTAINAB. PACE DT
0 LS Pt PIZZA TEAMS DT
0 Scrum Ds DAILY SCRUM DT
0 XP Co COLLECT. OWNER. DT
0 LS Ft FEATURE TEAMS DT
0 Scrum Sm SCRUM MASTER DT
3 Scrum Do DEFN. OF DONE DT
8 Scrum In PROD. INCREMENT DT
Development Team
©Jerome Kehrli / www.niceideas.ch
0 XP Sr SMALL RELEASES SMC
0 Scrum Es EST. STORY POINTS SMC
1 Scrum Po PRODUCT OWNER SMC
4 Scrum Tv TEAM VELOCITY SMC
4 Scrum Sp SPRINT SMC
0 XP Wt WHOLE TEAM SMC
0 Kanban Ko KANBAN BOARD SMC/PMC
1 Scrum Pp PLANNING POKER SMC
1 Scrum Sd SPRINT DEMO SMC
2 Scrum Sb SPRINT BACKLOG SMC
2 Scrum So SPRINT RETRO. SMC
4 DevOps Or OPERAT. RITUALS SMC
4 XP Pg PLANNING GAME SMC
5 Scrum Sl SPRINT PLANNING SMC
Sprint Management & Tools
0 XP Cs CODING STAND. AC
0 DAD Am ARCHITECT. MGMT AC
1 FDD Si SOLID PRINCIPLES AC
1 DevOps St SHARE THE TOOLS AC
2 Scrum Pb PROD. BACKLOG AC
3 XP Ac TESTS COVERAGE AC
4 XP Mt METAPHOR AC
6 XP Sn SIMPLE DESIGN AC
10 DevOps Cd CONTIN. DELIVERY AC
Archi-tecture
0 XP Oc ON-SITE CUSTOM. PMC
1 Kaizen Kb KAIZEN BURST SMC/PMC
0 DAD Pm PRODUCT MGMT PMC
1 DevOps Os OPERAT. STAKEH. PMC
2 LS Mv MVP PMC
2 LS Ab A/B TESTING PMC
2 XP At ACCEPTANCE TST. PMC
4 LS Fl FEEDBACK LOOP PMC
0 LS Gb GET OUT BUILDING PMC
1 Kaizen Wh 5 WHY's SMC/PMC
1 ProdDev Cc 3 C's PMC
1 ProdDev Pv PRODUCT VISION PMC
1 LS Fa FAIL FAST PMC
2 LS As ACTION. METRICS PMC
2 LS Bb BUILD VS. BUY AC
2 ProdDev Us USER STORIES PMC
3 ProdDev Iv I.N.V.E.S.T PMC
5 ProdDev Sg STORY MAPPING PMC
Product Management & Approach
Maturity / Difficuly
Principle / Ritual
Practice
X Method
Xy
PRACTICE NAME
XZ
X = complexity, expressed as the number of dependencies
Xy = Practice 2 letters Code
XZ = Related Team or Committee
eXtreme Program.
Scrum
DevOps
Product Develop.
Lean Startup
(Other)

3. Remarks

- Interestingly, but not surprisingly, scrum is really in the middle of the schematic, underlying the fact that it impacts as well development principles and the development team organization.
- XP is really everywhere down the line.
- Product Development is really about Product Management in the Agile world.
- DevOps is more related to development practices than everything else.

The next part of this article describes each and every principle and practice.

4. Principles and Practices

4.1 XP

Sn : Simple Design
A simple design always takes less time to finish than a complex one. So always do the simplest thing that could possibly work next. If you find something that is complex replace it with something simple. It's always faster and cheaper to replace complex code now, before you waste a lot more time on it.

Depends on Td, Rf, Su, Mt, Si, Am

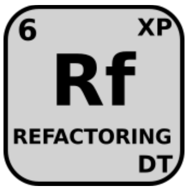
Mt : Metaphor
System Metaphor is itself a metaphor for a simple design with certain qualities. The most important quality is being able to explain the system design to new people without resorting to dumping huge documents on them. A design should have a structure that helps new people begin contributing quickly. The second quality is a design that makes naming classes and methods consistent.

Depends on Sn, Rf, Oc, Am

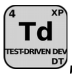
Td : TDD = Test Driven Development
Test-driven development is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only. This is opposed to software development that allows software to be added that is not proven to meet requirements.

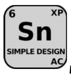
Depends on Wt, Ci, Cs, Sc

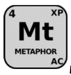
Oc : Onsite Customer
One of the few requirements of extreme programming (XP) is to have the customer available. Not only to help the development team, but to be a part of it as well. All phases of an XP project require communication with the customer, preferably face to face, on site. It's best to simply assign one or more customers to the development team.

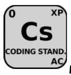


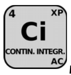
Rf : Refactoring
We computer programmers hold onto our software designs long after they have become unwieldy. We continue to use and reuse code that is no longer maintainable because it still works in some way and we are afraid to modify it. But is it really cost effective to do so? Extreme Programming (XP) takes the stance that it is not. When we remove redundancy, eliminate unused functionality, and rejuvenate obsolete designs we are refactoring. Refactoring throughout the entire project life cycle saves time and increases quality. Refactor mercilessly to keep the design simple as you go and to avoid needless clutter and complexity. Keep your code clean and concise so it is easier to understand, modify, and extend

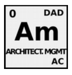
Depends on 

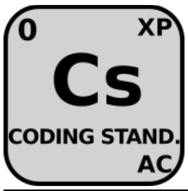
, 

, 

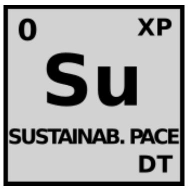
, 

, 

, 



Cs : Coding Standards
Code must be formatted to agreed coding standards. Coding standards keep the code consistent and easy for the entire team to read and refactor. Code that looks the same encourages collective ownership.



Su : Sustainable Pace
To set your pace you need to take your iteration ends seriously. You want the most completed, tested, integrated, production ready software you can get each iteration. Incomplete or buggy software represents an unknown amount of future effort, so you can't measure it. If it looks like you will not be able to get everything finished by iteration end have an iteration planning meeting and re-scope the iteration to maximize your project velocity. Even if there is only one day left in the iteration it is better to get the entire team re-focused on a single completed task than many incomplete ones.

0
XP
Wt
WHOLE TEAM
SMC

Wt : Whole Team
All the contributors to an XP project sit together, members of a whole team. The team shares the project goals and the responsibility for achieving them. This team must include a business representative, the "Customer" who provides the requirements, sets the priorities, and steers the project

Ci : Continuous Integration

Developers should be integrating and commiting code into the code repository every few hours, when ever possible. In any case never hold onto changes for more than a day. Continuous integration often avoids diverging or fragmented development efforts, where developers are not communicating with each other about what can be re-used, or what could be shared. Everyone needs to work with the latest version. Changes should not be made to obsolete code causing integration headaches.

Depends on Td, Cs, Co, Sc

Co : Collective Ownership

Collective Ownership encourages everyone to contribute new ideas to all segments of the project. Any developer can change any line of code to add functionality, fix bugs, improve designs or refactor. No one person becomes a bottle neck for changes.

Cr : Code Review

Code review is increasingly favored over strict Pair Programming as initially requires by the XP Method. The problem with Pair programming is that it cannot fitr everybody.

Code reviews are considered important by many large-process gurus. They are intended to ensure conformance to standards, and more importantly, intended to ensure that the code is clear, efficient, works, and has QWAN. They also intended to help disseminate knowledge about the code to the rest of the team.

Depends on Sn, Cs, Sc, Am

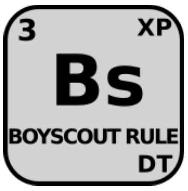
Pg : Planning Game

The main planning process within extreme programming is called the Planning Game. The game is a meeting that occurs once per iteration, typically once a week. The planning process is divided into two parts: Release Planning and Sprint Planning.

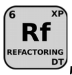
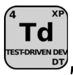
Depends on Sr, Oc, Po, Pm

Sr : Small Releases
The development team needs to release iterative versions of the system to the customers often. Some teams deploy new software into production every day. At the very least you will want to get new software into production every week or two. At the end of every iteration you will have tested, working, production ready software to demonstrate to your customers. The decision to put it into production is theirs.

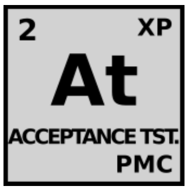
Sc : Source Code Management
A component of software configuration management, version control, also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information. Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.



Bs : Boyscout Rule
The Boy Scouts have a rule: "Always leave the campground cleaner than you found it." If you find a mess on the ground, you clean it up regardless of who might have made the mess. You intentionally improve the environment for the next group of campers. Actually the original form of that rule, written by Robert Stephenson Smyth Baden-Powell, the father of scouting, was "Try and leave this world a little better than you found it."
What if we followed a similar rule in our code: "Always check a module in cleaner than when you checked it out." No matter who the original author was, what if we always made some effort, no matter how small, to improve the module. What would be the result?

Depends on , ,
6
XP
Rf
REFACTORING
DT
4
XP
Td
TEST-DRIVEN DEV
DT
6
XP
Sn
SIMPLE DESIGN
AC

No : No premature optimization
In Donald Knuth's paper "Structured Programming With GoTo Statements", he wrote: "Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%."



At : Acceptance testing

Acceptance tests are created from user stories. During an iteration the user stories selected during the iteration planning meeting will be translated into acceptance tests. The customer specifies scenarios to test when a user story has been correctly implemented. A story can have one or many acceptance tests, what ever it takes to ensure the functionality works.

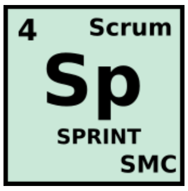
Depends on Oc, Sr

Ac : Automated Tests Coverage

Code Coverage is a measurement of how many lines/blocks/arcs of your code are executed while the automated tests are running.
Code coverage on every dimension should be above possible to 80% (the famous 80/20) rule and close to 100% (TDD).

Depends on Ci, Td, Am

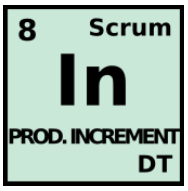
4.2 Scrum



Sp : Sprint

A Sprint is a time-box of one month or less during which a "Done", useable, and potentially releasable product Increment is created. Sprints best have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint.

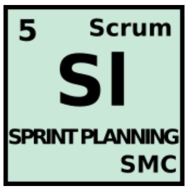
Depends on Sr, Sl, So, Sb



In : Product Increment (Shippable Product)

In Scrum, the Development Team delivers each Sprint a Product Increment. The increment must consist of thoroughly tested code that has been built into an executable, and the user operation of the functionality is documented either in Help files or user documentation. These requirements are documented in the Definition of Done.
If everything works fine and the Development Team has estimated well, the Product Increment includes all items, which were planned in the Sprint Backlog, tested and documented.

Depends on Sr, Sp, Pb, Ci, Cd, Ft, Pt, Ff

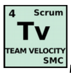


Sl : Sprint Planning

In Scrum, the sprint planning meeting is attended by the product owner, ScrumMaster and the entire Scrum team. Outside stakeholders may attend by invitation of the team, although this is rare in most companies.
During the sprint planning meeting, the product owner describes the highest priority features to the team. The team asks enough questions that they can turn a high-level user story of the product backlog into the more detailed tasks of the sprint backlog.

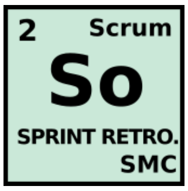
Depends on

,

, 

,

,

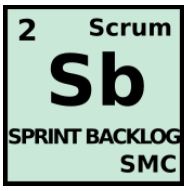


So : Sprint Retrospective

No matter how good a Scrum team is, there is always opportunity to improve. Although a good Scrum team will be constantly looking for improvement opportunities, the team should set aside a brief, dedicated period at the end of each sprint to deliberately reflect on how they are doing and to find ways to improve. This occurs during the sprint retrospective.
The sprint retrospective is usually the last thing done in a sprint. Many teams will do it immediately after the sprint review. The entire team, including both the ScrumMaster and the product owner should participate. You can schedule a scrum retrospective for up to an hour, which is usually quite sufficient. However, occasionally a hot topic will arise or a team conflict will escalate and the retrospective could take significantly longer.

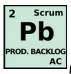
Depends on

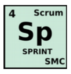
,

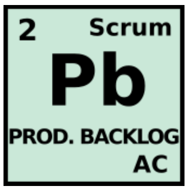


Sb : Sprint Backlog

The sprint backlog is a list of tasks identified by the Scrum team to be completed during the Scrum sprint. During the sprint planning meeting, the team selects some number of product backlog items, usually in the form of user stories, and identifies the tasks necessary to complete each user story. Most teams also estimate how many hours each task will take someone on the team to complete.

Depends on 

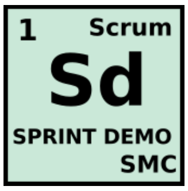
, 



Pb : Product Backlog

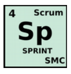
The agile product backlog in Scrum is a prioritized features list, containing short descriptions of all functionality desired in the product. When applying Scrum, it's not necessary to start a project with a lengthy, upfront effort to document all requirements. Typically, a Scrum team and its product owner begin by writing down everything they can think of for agile backlog priorization. This agile product backlog is almost always more than enough for a first sprint. The Scrum product backlog is then allowed to grow and change as more is learned about the product and its customers.

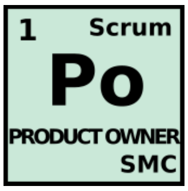
Depends on
5
ProdDev
Sg
STORY MAPPING
PMC
,
0
DAD
Pm
PRODUCT MGMT
AC



Sd : Sprint Demo

In Scrum, each sprint is required to deliver a potentially shippable product increment. This means that at the end of each sprint, the team has produced a coded, tested and usable piece of software.
So at the end of each sprint, a sprint review meeting is held. During this meeting, the Scrum team shows what they accomplished during the sprint. Typically this takes the form of a demo of the new features.

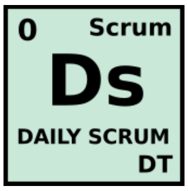
Depends on 



Po : Product Owner

The Scrum product owner is typically a project's key stakeholder. Part of the product owner responsibilities is to have a vision of what he or she wishes to build, and convey that vision to the scrum team. This is key to successfully starting any agile software development project. The agile product owner does this in part through the product backlog, which is a prioritized features list for the product.
The product owner is commonly a lead user of the system or someone from marketing, product management or anyone with a solid understanding of users, the market place, the competition and of future trends for the domain or type of system being developed.

Depends on
0
XP
Oc
ON-SITE CUSTOM.
PMC



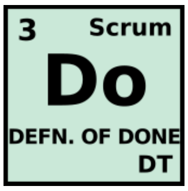
Ds : Daily Scrum

In Scrum, on each day of a sprint, the team holds a daily scrum meeting called the "daily scrum." Meetings are typically held in the same location and at the same time each day. Ideally, a daily scrum meeting is held in the morning, as it helps set the context for the coming day's work. These scrum meetings are strictly time-boxed to 15 minutes. This keeps the discussion brisk but relevant.

Sm : Scrum Master

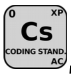
What is a Scrum Master? The ScrumMaster is responsible for making sure a Scrum team lives by the values and practices of Scrum. The ScrumMaster is often considered a coach for the team, helping the team do the best work it possibly can. The ScrumMaster can also be thought of as a process owner for the team, creating a balance with the project's key stakeholder, who is referred to as the product owner.

The ScrumMaster does anything possible to help the team perform at their highest level. This involves removing any impediments to progress, facilitating meetings, and doing things like working with the product owner to make sure the product backlog is in good shape and ready for the next sprint. The ScrumMaster role is commonly filled by a former project manager or a technical team leader but can be anyone.



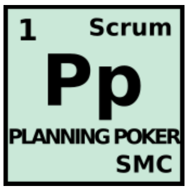
Do: Definition of Done

Definition of Done is a simple list of activities (writing code, coding comments, unit testing, integration testing, release notes, design documents, etc.) that add verifiable/demonstrable value to the product. Focusing on value-added steps allows the team to focus on what must be completed in order to build software while eliminating wasteful activities that only complicate software development efforts.

Depends on 

,

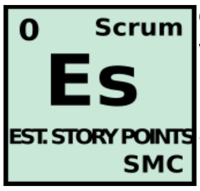
,



Pp : Planning Poker

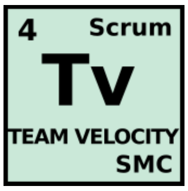
Planning Poker is an agile estimating and planning technique that is consensus based. To start a poker planning session, the product owner or customer reads an agile user story or describes a feature to the estimators. Each estimator is holding a deck of Planning Poker cards with values like 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100, which is the sequence we recommend. The values represent the number of story points, ideal days, or other units in which the team estimates.
The estimators discuss the feature, asking questions of the product owner as needed. When the feature has been fully discussed, each estimator privately selects one card to represent his or her estimate. All cards are then revealed at the same time.
If all estimators selected the same value, that becomes the estimate. If not, the estimators discuss their estimates. The high and low estimators should especially share their reasons. After further discussion, each estimator reselects an estimate card, and all cards are again revealed at the same time.
The poker planning process is repeated until consensus is achieved or until the estimators decide that agile estimating and planning of a particular item needs to be deferred until additional information can be acquired.

Depends on



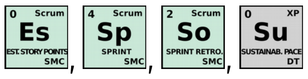
Es : Estimations in Story Points

Story points are a unit of measure for expressing an estimate of the overall effort that will be required to fully implement a product backlog item or any other piece of work.
When we estimate with story points, we assign a point value to each item. The raw values we assign are unimportant. What matters are the relative values. A story that is assigned a 2 should be twice as much as a story that is assigned a 1. It should also be two-thirds of a story that is estimated as 3 story points.
Instead of assigning 1, 2 and 3, that team could instead have assigned 100, 200 and 300. Or 1 million, 2 million and 3 million. It is the ratios that matter, not the actual numbers.

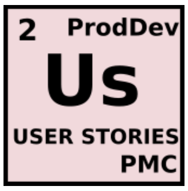


Tv : Team Velocity

Velocity is simply a metric based on the completed items in a sprint by a single team. The metric is completely subjective to that specific team, and should never be extrapolated for any other comparison.
Velocity is a reflective metric gathered from the sprint throughput of a stable team. Usually, a velocity metric is not considered valid until several sprints have been completed.

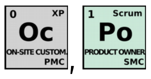
Depends on 
0 Scrum Es EST. STORY POINTS SMC
4 Scrum Sp SPRINT SMC
2 Scrum So SPRINT RETRO. SMC
0 XP Su SUSTAINAB. PACE DT

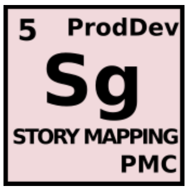
4.3 Product Development



Us : User Stories

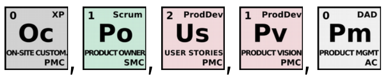
In software development and product management, a user story is an informal, natural language description of one or more features of a software system. User stories are often written from the perspective of an end user or user of a system. They are often recorded on index cards, on Post-it notes, or in project management software. Depending on the project, user stories may be written by various stakeholders including clients, users, managers or development team members.

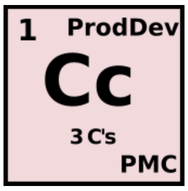
Depends on 



Sg : Story Mapping

Story mapping consists of ordering user stories along two independent dimensions. The "map" arranges user activities along the horizontal axis in rough order of priority (or "the order in which you would describe activities to explain the behaviour of the system"). Down the vertical axis, it represents increasing sophistication of the implementation.
Given a story map so arranged, the first horizontal row represents a "walking skeleton", a barebones but usable version of the product. Working through successive rows fleshes out the product with additional functionality.

Depends on 
0 XP Oc ON-SITE CUSTOM. PMC
1 Scrum Po PRODUCT OWNER SMC
2 ProdDev Us USER STORIES PMC
1 ProdDev Pv PRODUCT VISION PMC
0 DAD Pm PRODUCT MGMT AC



Cc : 3 C's - Card, conversation, confirmation

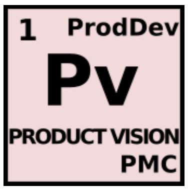
"Card, Conversation, Confirmation"; this formula (from Ron Jeffries) captures the components of a User Story:

a **"Card"** (or often a Post-It note), a physical token giving tangible and durable form to what would otherwise only be an abstraction;

a **"conversation"** taking place at different time and places during a project between the various people concerned by a given feature of a software product: customers, users, developers, testers; this conversation is largely verbal but most often supplemented by documentation;

the **"confirmation"**, finally, the more formal the better, that the objectives the conversation revolved around have been reached.

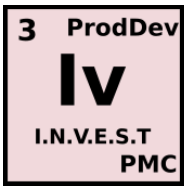
Depends on



Pv : Product Vision (elevator Pitch)

Every Scrum project needs a product vision that acts as the project's true north, sets the direction and guides the Scrum team. It is the overarching goal everyone must share – Product Owner, ScrumMaster, team, management, customers and other stakeholders. As Ken Schwaber puts it: "The minimum plan necessary to start a Scrum project consists of a vision and a Product Backlog. The vision describes why the project is being undertaken and what the desired end state is."

Depends on



Iv : INVEST

The INVEST mnemonic for agile software projects was created by Bill Wake as a reminder of the characteristics of a good quality User Story:

Independent: The User Story should be self-contained, in a way that there is no inherent dependency on another PBI;

Negotiable: User Stories are no contracts and must leave space for discussion;

Valuable: A User Story must deliver value to the stakeholders;

Estimatable: You must always be able to estimate the size of a User Story;

Small: User Stories should not be so big as to become impossible to plan/task/prioritize with a certain level of accuracy;

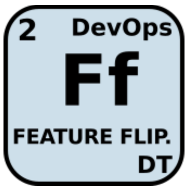
TestableThe User Story or its related description must provide the necessary information to make test development possible.

Depends on

,

,

4.4 DevOps



Ff : Feature Flipping

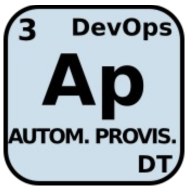
Feature flipping is a technique in software development that attempts to provide an alternative to maintaining multiple source-code branches (known as feature branches), such that the feature can be tested, even before it is completed and ready for release. Feature flipping is used to hide, enable or disable the features, during run time. For example, during the development process, the developer can enable the feature for testing and disable it for remaining users

Depends on Cm, Am

Cd : Continuous Delivery

Continuous delivery (CD) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time. It aims at building, testing, and releasing software faster and more frequently. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery.

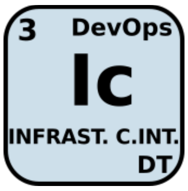
Depends on Ci, Td, Ap, In, Sr, Ic, Zd, Vc, Bp, Ar



Ap : Automated Provisioning

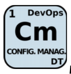
(Infrastructure as Code) Server provisioning is a set of actions to prepare a server with appropriate systems, data and software, and make it ready for network operation. Typical tasks when provisioning a server are: select a server from a pool of available servers, load the appropriate software (operating system, device drivers, middleware, and applications), appropriately customize and configure the system and the software to create or change a boot image for this server, and then change its parameters, such as IP address, IP Gateway to find associated network and storage resources (sometimes separated as resource provisioning) to audit the system
With DevOps and Automated Provisioning, this whole configuration pipeline should be completely automated and executable in one-click, either automatically or on-demand.

Depends on Ic, Cm, Vc

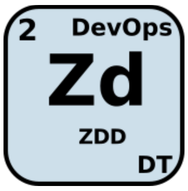


Ic : Infrastructure Continuous Integration
(Infrastructure as Code) Infrastructure Continuous Integration consists in leveraging Continuous Integration techniques to Infrastructure components. The continuous integration system is necessarily complex, spanning the development, test and staging environments. The continuous integration build should continuously build and test the provisioning, configuring and maintaining of the various infrastructure components.

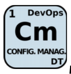
Depends on

, 

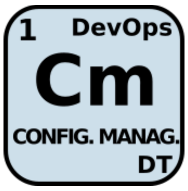
,



Zd : Zero Downtime Deployments
A Zero Downtime Deployment consists in redeploying (typically for a software upgrade) a production system without any downtime appearing to end users. To achieve such lofty goals, redundancy becomes a critical requirement at every level of your infrastructure. There are various techniques involved such a canari release or blue-green deployments.

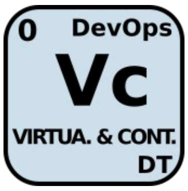
Depends on 

,



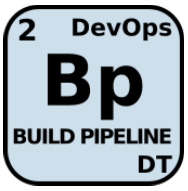
Cm : Configuration Management
Configuration management is a class of tool supporting the automation of the configuration of a system, platform or software. It typically consists in *define-with-code* the various config elements that prepare a provisioned compute resource (like a server or AWS Ec2 instance) for service (installing software, setting up users, configuring services, placing files with template-defined variables, defining external config resources like DNS records in a relevant zone).

Depends on



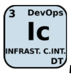
Vc : Virtualization and Containers

Hardware virtualization or platform virtualization refers to the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines is separated from the underlying hardware resources.
Containerization - also called container-based virtualization and application containerization - is an OS-level virtualization method for deploying and running distributed applications without launching an entire VM for each application. Instead, multiple isolated systems, called containers, are run on a single control host and access a single kernel.

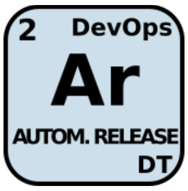


Bp : Build Pipelines

Build pipelines are integrated views of downstream and upstream build jobs on a build server. Build pipelines are requires to automated all the various tasks towards continuous delivery such as : provisionning of the environment, build of the various software (with compilation, tests, packaging, etc.), deployment of the software components, applying configuration and testing the deployed platform.

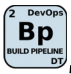
Depends on 

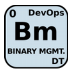
,
4
XP
Ci
CONTIN. INTEGR.
AC



Ar : Automated Releases

Release Automation consists in automating all the various steps requiered to release a new version of a software: building, testing, tagging, branching et depliying the binaries to a Binary management tools.

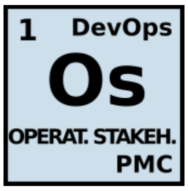
Depends on 

, 

St : Share the tools

Share the tools is a DevOps principles aimed at leveraging both Dev and Ops tools and practices to the other side of the wall. Developers should leverage their automation and building tool to Infrastructure Automation, Provisioning and Testing. Ops should share the production monitoring concerns with developers.

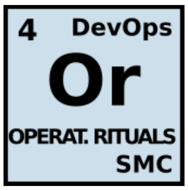
Depends on
0
DAD
Am
ARCHITECT. MGMT
AC



Os : Operators are stakeholders

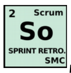
Operators as stakeholders is a DevOps principle stating that Operators should be considered the other users of the platform. They should be fully integrated in the Software Development Process.
At specification time, operators should give their non-functional requirements just as business users give their functional requirement. Such non-functional requirements should be handled with same important and priority by the development team.
At implementation time, operators should provide feedback and non-functional tests specifications continuously just as business users provides feedback on functional features.

Depends on
0
DAD
Pm
PRODUCT MGMT
AC

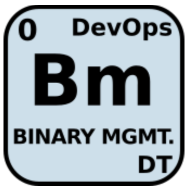


Or : Operators in Rituals

Operators in Rituals is a DevOps principle stating that operators should be integrated in the Development Team Rituals such as the Sprint Planning and Sprint Retrospective and represent non-functional constraints during these rituals just as the Product Owner represents the functional interests.

Depends on
5
Scrum
Sl
SPRINT PLANNING
SMC
, 

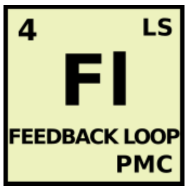
,
0
Scrum
Ds
DAILY SCRUM
DT
,
10
DevOps
Cd
CONTIN. DELIVERY
AC



Bm : Binaries Management

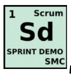
A binary repository manager is a software tool designed to optimize the download and storage of binary files used and produced in software development. It centralizes the management of all the binary artifacts generated and used by the organization to overcome the complexity arising from the diversity of binary artifact types, their position in the overall workflow and the dependencies between them.
A binary repository is a software repository for packages, artifacts and their corresponding metadata. It can be used to store binary files produced by an organization itself, such as product releases and nightly product builds, or for third party binaries which must be treated differently for both technical and legal reasons.

4.5 Lean Startup



Fl : Feedback Loop

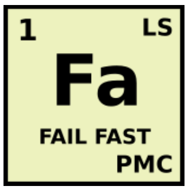
The *Build-Measure-Learn* feedback loop is one of the central principles of Lean Startup Method.
A startup is to find a successful revenue model that can be developed with further investment. Build-Measure-Learn is a framework for establishing – and continuously improving – the effectiveness of new products, services and ideas quickly and cost-effectively.
In practice, the model involves a cycle of creating and testing hypotheses by building something small for potential customers to try, measuring their reactions, and learning from the results.

Depends on , , ,

---

Ft : feature Teams

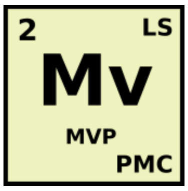
A *feature team* is a long-lived, cross-functional, cross-component team that completes many end-to-end customer features—one by one. It is opposed to the traditional approach of *Component Team* where a team is specialized on an individual software components and maintains it over several projects at the same time.
The Feature team approach seeks to avoid the bottlenecks usually appearing with Component Teams.



Fa : Fail Fast

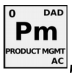
Fail fast means getting out of planning mode and into testing mode, eventually for every critical component of your model of change. Customer development is the process that embodies this principle and helps you determine which hypotheses to start with and which are the most critical for your new idea.
An important goal of the philosophy is to cut losses when testing reveals something isn't working and quickly try something else, a concept known as pivoting.

Depends on

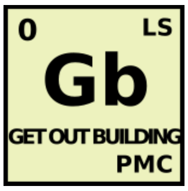


Mv : MVP

In product development, the minimum viable product (MVP) is a product with just enough features to satisfy early customers, and to provide feedback for future development

Depends on 

,
0
DAD
Am
ARCHITECT. MGMT
AC



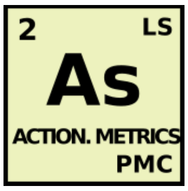
Gb : Get Out of the building

If you are pre-Product/Market Fit and you aren't actually "Getting out of the Building" (actually talking to your customers), you aren't doing Customer Development, and your startup isn't a Lean Startup.
Again: If you aren't actually talking to your customers, you aren't doing Customer Development.

0
LS
Pt
PIZZA TEAMS
DT

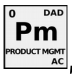
Pt : Pizza Teams

The idea of a "two pizza team" was coined by Jeff Bezo, founder of Amazon.com. If you can't feed a team with two pizzas, it's too large. That limits a task force to five to seven people, depending on their appetites."
The underlying idea is that as a team's size grows, the amount of one-on-one communication channels tend to explode.
Beyond ten, communication loses efficiency, cohesion diminishes, parasitism behaviors and power struggles appear, and the performance of the team decreases very rapidly with the number of members.

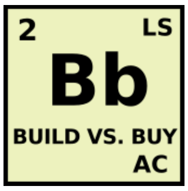


As : Actionable Metrics

The only metrics that entrepreneurs should invest energy in collecting are those that help them make decisions. Actionable Metrics are opposed to Vanity Metrics.
This is a precision of another fundamental Lean Startup practice wich is "Obsession of Measure" stating that everything should be measured and no decision should be taken in the company if it is not supported by a Key Process Indicator or a Key Risk Indicator.

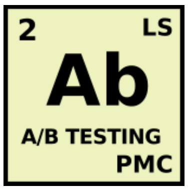
Depends on 

,
0
DAD
Am
ARCHITECT. MGMT
AC



Bb : Build vs. Buy
This is a fundamental principle of the Lean Startup and the web giants : favor as much as possible building your own software, your own feature instead of buying a third party software or library.
When initiating a startup, having to pay fees to third party corporations before reaching a sustainable growth is suicidal.

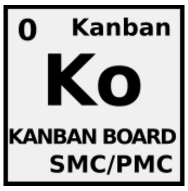
Depends on Pm, Am



Ab : A/B Testing
In marketing and business intelligence, A/B testing is a term for a controlled experiment with two variants, A and B. It can be considered as a form of statistical hypothesis testing with two variants leading to the technical term, two-sample hypothesis testing, used in the field of statistics

Depends on Pm, Am

4.6 Kanban



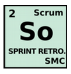
Ko : Kanban Board
A Kanban board is a work and workflow visualization tool that enables you to optimize the flow of your work. Physical Kanban boards typically use sticky notes on a whiteboard to communicate status, progress, and issues.
An agile corporation should use a KanBan board to monitor all its processes.
A development team will typically use a Kanban board to monitor the Sprint backlog completion during a sprint.

4.7 Kaizen

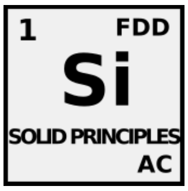
Kb : Kaizen Burst
The Kaizen burst is a specific Kaizen process integrated the the development rituals. In Agile Software Development, it is really integrated in the Sprint Retrospective. This idea is to identify in a visual way (with a post-it on a board for instance) the weaknesses or problems in the development practices or processes. These boxes are called Kaizen burst.
Theses boxes are commented as actions are taken towards improvement and eventuelly removed when the weakness has been adressed or the problem solved.

Depends on So

Wh : 5 Why
5 Whys is an iterative interrogative technique used to explore the cause-and-effect relationships underlying a particular problem.
The primary goal of the technique is to determine the root cause of a defect or problem by repeating the question "Why?" Each answer forms the basis of the next question. The "5" in the name derives from an anecdotal observation on the number of iterations needed to resolve the problem.

Depends on 

4.8 FDD (Feature Driven Development)



Si : SOLID principles
In computer programming, the term SOLID is a mnemonic acronym for five design principles intended to make software designs more understandable, flexible and maintainable. The principles are a subset of many princples promoted by Robert C. Martin.
Though they apply to any object-oriented design, the SOLID principles can also form a core philosophy for methodologies such as agile development or Adaptive Software Development.
The 5 principles are as follows:

SRP : Single responsibility principle - a class should have only a single responsibility (i.e. only one potential change in the software's specification should be able to affect the specification of the class)

OCP : Open/closed principle - "software entities ... should be open for extension, but closed for modification."

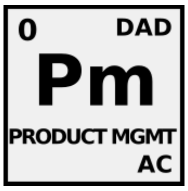
LSP : Liskov substitution principle - "objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program."

ISP : Interface segregation principle - "many client-specific interfaces are better than one general-purpose interface."

DIP : Dependency inversion principle - one should "depend upon abstractions, not concretions."

Depends on

4.9 DAD

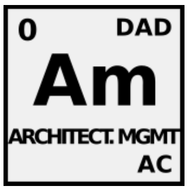


Pm : Product Management Committee

The Product Management Committee is both a team and a ritual that enforces a smart approach to product management.

Product Management consists in identifying and evolving your organization’s business vision; identifying and prioritizing potential products/solutions to support that vision; identifying, prioritizing, and allocating features to products under development; managing functional dependencies between products; and marketing those products to their potential customers.

The Product Management Committee is the weekly (or bi-weekly) ritual enforcing and supporting this process with the required role attending the committee. It is led by the product Owner which has more a role of facilitator and arbitrator that a formal decision role. The Product Owner represents the PMC to the development team.



Am : Architecture Committee

The Architecture Committee is responsible to analyze user stories and define Development Tasks. Every story should be specified, designed and discussed. Screen mockups if applicable should be drawn, acceptance criteria agreed, etc.

Since the Architecture Committee is also responsible for estimating Stories, it's important that representatives of the Development Team, not only the Tech Leads and the Architects, but simple developers as well, take part in it.